NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4032 : Data Analytics and Mining

Group 10

Kenneth Lee U1922011L

Uday Nihal Arya U2023794

Bryan Goh Zheng Ting U1922436E

Bharat Manikkath U2022906E

Liew Hon Weng U1922283F

Krishna Shivangi U2023035E

Review of Clustering Algorithms

Abstract

Clustering algorithms have been developed as Unsupervised Machine Learning models with the purpose of creating groups of data points such that data points within each group are more similar to each other than to points belonging to other groups. This helps to identify patterns within the data, flag outliers and even improve the performance of Supervised Machine Learning models by giving greater insight into the structure of the data.

There now exists a vast variety of clustering algorithms and significantly improved implementations of basic clustering algorithms. Clustering algorithms have also been categorized into different types in the literature. This report aims to compare the performance of clustering algorithms belonging to different categories on real-world datasets.

This report selects existing real-world datasets with different properties and examines the performance of various candidate clustering algorithms from different categories. We aim to use empirical metrics for comparison, while also explaining the results based on a theoretical understanding of the algorithms. Overall, we intend to promote discussion on the suitability of a clustering algorithm with respect to the structure of the database it is applied on.

1. Introduction

1.1. Background

Data clustering has been used in a broad spectrum of applications, from image segmentation to vector and colour image quantization, customer segmentation, data mining, compression and further learning for machine learning [1]. It is essentially the process of dividing a set of data points into groups or clusters within multidimensional data based on some form of similarity measure. [2].

There are many types of similarity measures used to identify how similar two data points are to each other. Such measures include Euclidean distance, Manhattan distance, Hamming distance, cosine similarity and many more. There are a few categories of clustering algorithms: partition-based, hierarchical-based, density-based and grid-based. [3]

Many studies have been done on the different categories, and improvements have been suggested to improve the algorithms in each of those categories. One of the main goals of clustering improvement has been to create higher-quality clusters with high intra-class similarity and low inter-class similarity based on a predefined similarity measure. As the quality of the clusters is largely due to the similarity measure used by the method and its implementation, much research has been done to improve similarity measures and implementation to improve clustering between similar points. [4]

In essence, it is a collection of objects that are similar and dissimilar. In general, it is used to identify meaningful structure, explanatory underlying processes, generative features and groupings.

1.2 Project Overview

To perform clustering, we've used different Clustering Algorithms such as

- K-means
- Agglomerative Clustering (Linkage methods - Single, Complete, Average, Ward)
- DBSCAN
- Gaussian Mixture

In this project, we implemented the above-mentioned clustering algorithms on four different datasets:

- Customer spending dataset (Source: Kaggle : Hierarchical Clustering for Customer Data | Kaggle)
- Rock dataset (Source: UCI Repository : UCI Machine Learning Repository: Connectionist Bench (Sonar, Mines vs. Rocks) Data Set)
- Wine dataset (Source: UCI Repository : UCI Machine Learning Repository: Wine Data Set)
- Yeast dataset (Source: UCI Repository : UCI Machine Learning Repository: Yeast Data Set)

The aim of this project was to understand the strengths and weaknesses of the different clustering algorithms and how it affected the clustering of our data points with respect to the different types of datasets that we have used.

We compared the accuracy of the different clustering algorithms such as:

- Silhouette Score
- Davies-Bouldin Index
- Calinski Harabasz Score
- Classification Accuracy (for those datasets which had a defined cluster label before implementing clustering)

2. Literature Review

2.1 Similarity Measures

2.1.1 Euclidean Distance

The euclidean distance is the length of a line segment between the two points. It is the square root of the sum of the squared differences.

EuclideanDistance : $d(x, y) = \sqrt{\sum_{i}^{N} (x_i - y_i)^2}$

Euclidean Distance is the most widely used distance metric as it is the default metric in the Sklearn library.

2.1.2 Hamming Distance

Hamming distance is the measure of the number of positions at which two corresponding pieces of string of the same length are different. It is measured by calculating the sum of the number of positions at which the two strings are different

HammingDistance : $D_H(x, y) = \sum_{i}^{N}(|x_i - y_i|)$

$$x = y \Rightarrow D = 0$$
$$x \neq y \Rightarrow D = 1$$

To get the average bit difference:

Average HammingDistance : $D_{H_{avg}}(x, y) = \frac{1}{N}\sum_{i}^{N}(|x_i - y_i|)$

2.1.3 Manhattan Distance

The Manhattan Distance between two points is the distance between two points measured along the axes at right angles

Manhattan Distance: $d(x, y) = \sum_{i}^{N} |x_i - y_i|$

2.1.4 Cosine Distance

The cosine distance between two points is used to see how similar they are. The sequences are viewed as vectors in an inner product space, and the cosine similarity is defined as the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths.

Cosine Distance: $cos(x, y) = \frac{x \cdot y}{||x|| \, ||y||}$

It is generally used as a metric for measuring distance when the magnitude of the vectors does not matter.

2.1.5 Mahalanobis Distance

The Mahalanobis distance measures distance relative to the centroid [5].

Mahalanobis Distance: $D^2 = (x - m)^T \cdot C^{-1} \cdot (x - m)$

D^2	: is the squared Mahalanobis distance
x	: is the vector of the observation (row in a dataset)
m	: is the vector of mean values of independent variables (mean of each column)
C^{-1}	: is the inverse covariance matrix of independent variables

It accounts for the variance of each variable and the covariance between variables by transforming the data into standardized uncorrelated data and computing the ordinary Euclidean distance for the transformed data from the point to the centroid. The Mahalanobis distance is like a univariate z-score that provides a way to measure distances that take into account the scale of the data, unlike ordinary Euclidean distance measured between two points.

The most common use for the Mahalanobis distance is to find multivariate outliers, which indicate unusual combinations of two or more variables [6].

2.2 Evaluation Scores

2.2.1 Silhouette Score

Silhouette score is a metric used to calculate how good the clustering technique is by looking at the consistency within the clusters [7]. Silhouette scoring is mainly used for distance based clustering algorithms. The values for this score range from -1 to 1, where:

- A value of 1 means : Clusters are well apart from each other and they are clearly distinguished
- A value of 0 means : The distance between the clusters is insignificant and there is overlapping
- A value of -1 means : The clusters are assigned the wrong way

Silhouette score, however, has a very high computational complexity.

2.2.2 Davies-Bouldin Index

Davies Bouldin index is a clustering evaluation measure. It is calculated as the average similarity of each cluster with a cluster is it most similar to, where similarity is the ratio of within cluster distances [8]. Its minimum value is 0. The lower the average similarity is, the better the clusters are separated and the better the clusters have split .

Using this index however, limits the distance metric to Euclidean space

2.2.3 Calinski Harabasz Score

Calinski Harabasz score is the ratio of the sum of cluster dispersion and inter cluster dispersion for all the clusters. The higher the score, the better the performance is [9]. This high score is achieved when the clusters are dense and well separated from each other .

2.2.4 Classification Accuracy

We devised a way to find the classification accuracy by writing a function that calculates how well our clustering algorithms correctly cluster the data points based on the preset labels that were present in the data.

eg :- yeast.data contains 10 clusters with unique string labels for the 10 different clusters. We want to see how well our clustering algorithm is able to cluster the data points of the same predefined clusters (points with the same label) together by treating it as a classification problem.

The steps are as follows -

- The function creates a list of labels present in the data, sorted based on the number of data points that belong to each label.
- The function then creates a list of predicted clusters (cluster number output from our algorithm), sorted based on the number of data points that belong to the predicted clusters.
- A mapping is done between the true label and the predicted clusters based on the two lists above. The true label with the highest number of data points will be mapped to the predicted cluster with the highest number of data points and this is done for the second highest, third highest uptil the n-th cluster.
- With this mapping we map the predicted clusters (cluster number output from our algorithm) to the predicted labels (actual true labels present in our original data).
- We sum up the number of data points that were predicted correctly (true label matches predicted label).
- We calculate the accuracy by dividing the sum from the previous step by the length of our dataset.

For DBSCAN the function was slightly modified since we cannot set the number of clusters in DBSCAN. It takes into account that there is a different number of actual clusters when compared to the number of clusters outputted by DBSCAN.

2.3 Curse of Dimensionality

When working with high dimensional data (where there are many attributes/columns), clustering algorithms typically tend to fall short in both analysis of such data, as well as visualization of these data [10]. The reason why some clustering algorithms fall short when dealing with high dimensional data could in part be due to the distance function that it uses to calculate the distance between two points. An example could be the euclidean distance which measures the distance between vectors, as each new attribute adds a non-negative term to the sum total of distance, the distance will increase as the number of dimensions increases.

To reduce dimensions for datasets, we could either use feature elimination methods or feature extraction methods such as Principal Component Analysis and t-Distributed Stochastic Neighbor Embedding (t-SNE)

2.4 Principal Component Analysis

One possible way of overcoming the curse of dimensionality in both analysis and visualization would be to use Principal Component Analysis (PCA) to reduce the dimensions of the data. PCA is a linear dimensionality reduction algorithm that finds new variables/attributes called Principal Component that are linear functions of those in the original dataset [11]. It tries to maximize variance and reduce noise while at the same time tries to minimize information loss and it reduces to solving an eigenvector problem [12]

At the end, we will obtain a certain number of principal components, and the first Principal Component explains the maximum variance, then the second Principal Component explains the variance not explained by the first Principal Component, and each principal component proceeds to explain the remaining variance not explained by its preceding component.

2.5 Basic Explanation of Clustering Categories Used

2.5.1 Partition-based Clustering

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one cluster. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. There is one important parameter to be considered for K-means, the number of clusters - K [13]. The algorithm starts by randomly selecting a centroid for each cluster, the number of centroids, K, having already been initialized. Once the centroids have all been initialized, each point is assigned to the closest cluster centroid. Once all points are assigned to either cluster we calculate the new centroids of the newly formed clusters. This process of assigning points and calculating the new centroid continues until the centroids of newly formed clusters do not change or the maximum number of iterations has been reached.

2.5.2 Hierarchical-Based Clustering

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from the other, and the objects within each cluster are broadly similar to each other [14].

Agglomerative - Agglomerative clustering is a sub-type of hierarchical clustering that follows a bottom-up approach that starts from an individual cluster (a leaf) and the distance between each cluster will be calculated. Clusters with the shortest distance will merge, creating a node. Distance between clusters will be calculated again before the merging continues. Different linkage methods can be used when calculating the distance between clusters during the merging process.

2.5.3 Density-Based Clustering

DBSCAN - Density-Based spatial clustering of applications with noise. It is a type of density-based clustering that groups clusters based on their density (how close points are to each other).
There are two important parameters to be considered for DBSCAN, and they are the epsilon_radius and minPoints.
The algorithm tries to identify three different types of points. Core points, Border points and Outliers. A core point is defined if there are at least minPoints surrounding its area with radius epsilon. A border point is defined if it is not a core point and is directly reachable from the core point. Outliers are points that are not reachable from any core point. A cluster defined by DBSCAN would then be constructed by merging reachable core points and border points.

2.5.4 Model-Based Clustering

Model-based clustering is a statistical approach to data clustering. The observed data is assumed to have been generated from a finite mixture of component models. Each component model is a probability distribution, typically a parametric multivariate distribution. For example, in a multivariate Gaussian mixture model, each component is a multivariate Gaussian distribution. The component responsible for generating a particular observation determines the cluster to which the observation belongs. However, the component generating each observation as well as the parameters for each of the component distributions are unknown. The key learning task is to determine the component responsible for generating each observation, which in turn gives the clustering of the data. Ideally, observations generated from the same component are inferred to belong to the same cluster.[20]

3. Methodology

3.1 Dataset and Pre-processing

Data pre-processing:

- Checked and removed all rows that had null/NaN data so that we could fit our model on all features.
- We removed the label column (when predefined labels were present) before applying our clustering algorithms. The label column was used later to find the classification accuracy.
- Standardized the datasets to reduce the impacts of anomalies on the model and to prevent our models which depend on distance (Kmeans, Agglomerative and DBSCAN) from being biased by one single feature with large values. By scaling our data, all the features would contribute equally while clustering.
- Applied Principal Component Analysis (PCA) on our dataset (Ran the algorithms without PCA once to see the effects that PCA has on the algorithms)

Examples of the data pre-processing can be found in the appendix below.

3.1.1 Customer Spending dataset

This dataset contains 5 different columns and 200 rows of data -

The dataset was standardized.

We chose this dataset as it was the only dataset we had which was not a classification problem. This dataset would help us compare how our clustering algorithms work without having a fixed number of clusters and also compare how many clusters they would

3.1.2 Sonar dataset

This dataset contains 61 different columns and 208 rows of data -

The last column was removed as it was a column used to label the datasets and the dataset was standardized.

We chose this dataset due to the high number of columns. This dataset would help us compare which one of our four clustering algorithms works best with data that has high dimensionality and how high dimensionality affects the different accuracy metrics for our four clustering algorithms.

3.1.3 Yeast dataset

This dataset contains 10 different columns and 1479 rows of data -

The last column was removed as it was a column used to label the datasets. The dataset was standardized.

We chose this dataset because it has multiple entries (1479 rows). The yeast dataset was the largest dataset in comparison to the others. This dataset would help us compare which of our four clustering algorithms works best and clusters the data as precisely as possible when there are many data points.

3.1.4 Wine dataset

This dataset contains 14 different columns and 178 rows of data -

The first column was removed as it was a column used to label the datasets. The dataset was standardized.

We chose this dataset because it is small and simple. The wine dataset is to be used as our control dataset where we explore the different clustering algorithms, without extreme sizes or complexity in the data set.

3.2 Clustering Algorithm

3.2.1 KMeans Clustering

Kmeans clustering is a centroid-based or distance-based algorithm, where we calculate the distance between points and centroids in order to assign them to clusters. [15]
For this project, we used the Scikit-learn library to implement the KMeans function.

The algorithm starts by randomly selecting centroids, which are used as starting points for each cluster. The algorithm then carries out an iterative process of assigning the data points to the ideal cluster and optimizing the positions of the centroids [15].
Steps for KMeans clustering:

1. Choosing the number of clusters
    a. 3 of the datasets chosen (yeast, wine and sonar) for this project had a correct classification column. We found the number of unique categories and labelled that as the number of clusters. For the customer data set, there was no correct classification column therefore we plot the elbow curve to obtain the optimal number of clusters as shown in Figure 1. This method consists of plotting the explained variation as a function of the number of clusters. Picking the elbow of the curve gives us the optimal cluster number which can be seen as 6 from the graph.

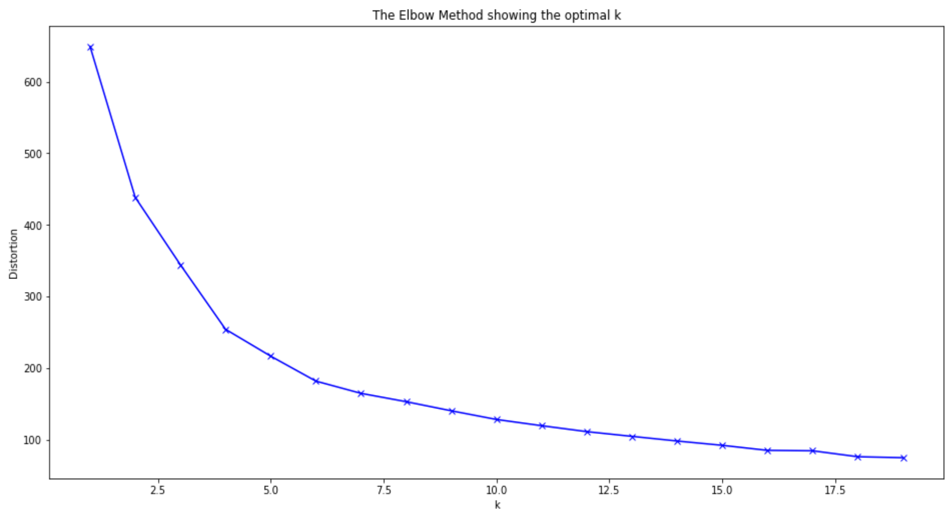


Figure 1

2. Select k random points from the data as centroids:
    a. For KMeans we randomly select centroids for each cluster for the algorithm
    b. For further experimenting on KMeans, we also implemented KMeans++ for comparing performances between the two algorithms. For KMeans++ we initialized on centroids before starting the Kmeans algorithm.
3. Assign data points to the closest cluster;
    a. The KMeans function will perform the calculations and assign data points to a cluster. The distance between all the data points and centroids will be calculated, and the data point to the closest centroid will be assigned to that cluster.
4. Updating centroids:
    a. As centroids were chosen randomly at the start of the algorithm in the case for KMeans, the centroid value will now be updated by finding the average of all the data points within the cluster.
    b. For KMeans++, even though we initialized the centroids from the start, we still need to update the centroid after each iteration

Steps 3 and 4 are performed iteratively.

5. The algorithms stops when:
    a. The centroids have been stabilized, meaning there is no change in the centroid value because the clustering process was successful
    b. Data points assigned to the clusters dont change
    c. The defined number of iterations has been completed

When evaluating both the clustering algorithms we see that the overall performance of KMeans++ is better than that of KMeans for the yeast, sonar and wine datasets. For the customer dataset, KMeans outperformed KMeans++ for all the evaluation scores.

Advantages of KMeans:

- Simple to implement
- Guarantees convergence
- Generalizes to clusters of different shapes and sizes

Disadvantages of KMeans:

- Having to choose a number of clusters manually if right classification for dataset is not given
- Centroids can be influenced by outliers
- Difficult to scale with a number of dimensions. Have to perform PCA
- Trouble clustering data where clusters are of varying sizes and density [16]

3.2.2 Agglomerative Clustering

Agglomerative Clustering is a hierarchy clustering procedure that involves dividing or agglomerating the original entities into smaller subgroups based on some criteria. [2] Therefore, hierarchical methods can provide valuable information about the interrelationships between categories, which is why they are used in a variety of fields. [3] Hierarchical clustering methods may construct the hierarchy in two opposite directions, bottom-up (agglomerative) and top-down (divisive). [4] In the case of our implementation, we used the bottom-up (agglomerative) approach. The agglomerating criteria chosen redefines the hierarchical clustering method. [4] In our implementation of agglomerative clustering, we used four different criterias - Single-Linkage, Complete-Linkage, Average-Linkage and Ward-Linkage to decide which data point belongs in which dataset. Single-linkage is the method in which the distance between two clusters is the minimum distance between the data points in the clusters. Complete-linkage is the method in which the distance between two clusters is the maximum distance between the data points in the clusters. Average-linkage is the method in which the distance between two clusters is the average distance of all the data points in the clusters. [4] Ward-linkage is the method in which it joins clusters that leads to the minimum increase in within-cluster variance. [5]
For our implementation, we used the scikit-learn API for Agglomerative Clustering to perform the clustering for our datasets.We used the default euclidean distance as the to measure the distance between the different features of our data points.
Based on our implementation of the Agglomerative Clustering algorithm using the the four different linkage methods, we can conclude the following:
(Refer to the graphs and table of scores in Appendix B below)

- Shortest time taken to run the algorithm: Average-linkage
- Best Silhouette Score: No definitive conclusion
- Best Davies Bouldin Score: Complete-linkage or Ward-linkage
- Calinski Harabasz Score: Ward-linkage
- Classification Accuracy: Single-linkage or Average-linkage

Advantages of Agglomerative Clustering:

- It can produce an ordering of objects, which may be informative when displaying results.
- Smaller clusters will be created during the clustering process, which may allow us to discover similarity in our data.

Disadvantages of Agglomerative Clustering:

- Sensitive to noise and outliers
    - As agglomerative clustering calculates the distance between 2 clusters before deciding on which 2 to merge, outliers will greatly affect clustering
- Not good for large data sets due to high time and space complexity
    - The space required for the Hierarchical clustering Technique is very high when the number of data points are high as we need to store the similarity matrix in the RAM. Space complexity is O(n^2)
    - Time complexity is O(n^3) [14]
- Unable to backtrack

3.2.3 DBSCAN Clustering

DBSCAN is a density based clustering algorithm and clusters are found by recursively taking core points of high density, finding its neighbors that are core samples, then finding all of the neighbors of those core points and until it reaches a border point and no further core point can be expanded.
For implementation, we used scikit-learn API for DBSCAN to do clustering for the four datasets. For all datasets, we have also used the default euclidean distance in scikit-learn to measure the distance between features of the data. After processing the data and fitting it into the algorithm, the algorithm will return its labels along with some other information. In scikit-learn, the noisy points are labeled as -1 and the rest of the clusters are labeled numerically 0,1,2 and so on [18].
As mentioned above, two key parameters for the algorithms are the epsilon radius (eps) which considers how close two points should be to be considered neighbors and the min samples which defines the minimum number of surrounding points for a point to be considered a core point [17].

The min samples are useful for setting how tolerant the algorithm is towards noise, and if the dataset is very noisy, a larger min sample value could be considered to identify the noise points.
On the other hand, identifying the eps is critical for the performance of the algorithm. Too small a value for eps will result in clusters not being formed while a large eps value causes the entire data sets to be returned as a single cluster, the value of epsilon directly impacts the quality of clusters found. In Figure 2, an experiment was done on the wine dataset to see the effects of different epsilon values. In this case, the optimal value of epsilon in this case would be in the range of 0.500. As can be seen from the experiments, too high an epsilon value (=1) will cause the entire dataset to be considered one cluster, while too low a value (0.1) will cause each point to be considered a cluster.

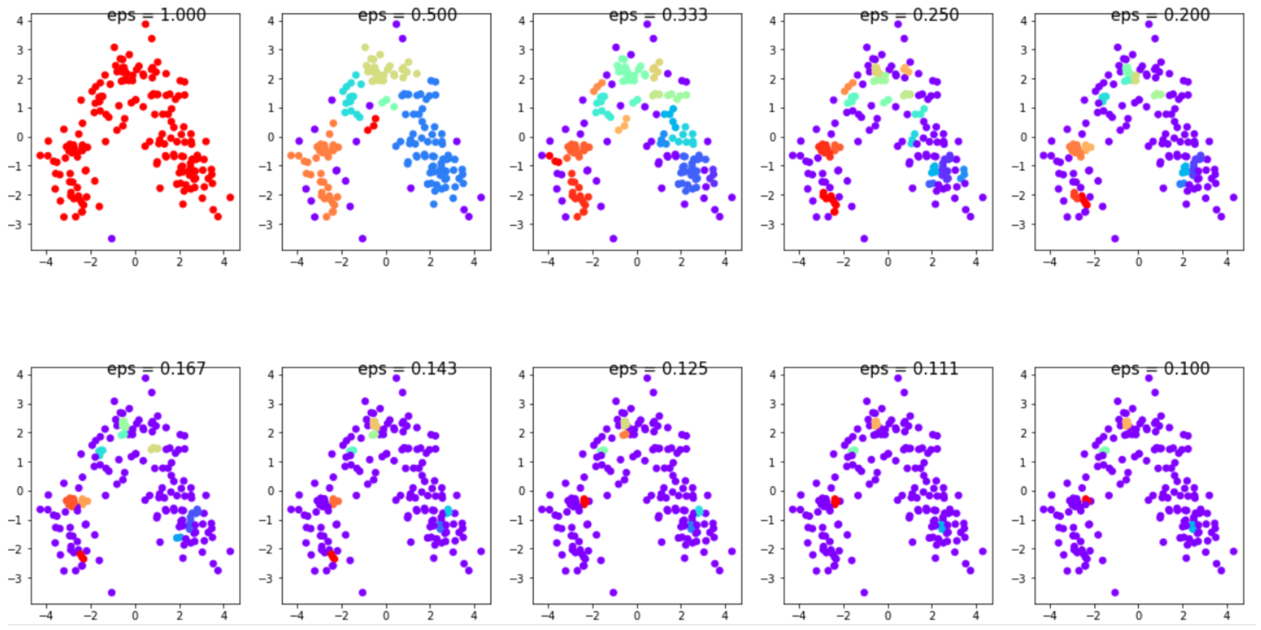


Figure 2

Therefore, one heuristic that we have implemented was to use a nearest neighbor distance plot to identify the elbow point and choose the eps value close to the point. As shown in Figure 3, we have plotted out the 3rd-NN distance, and it shows for most data points, the 3rd-NN is within the elbow point, roughly (eps = 2.4~3). This will ensure that most data points generated from DBSCAN would be core points, creating well defined clusters.

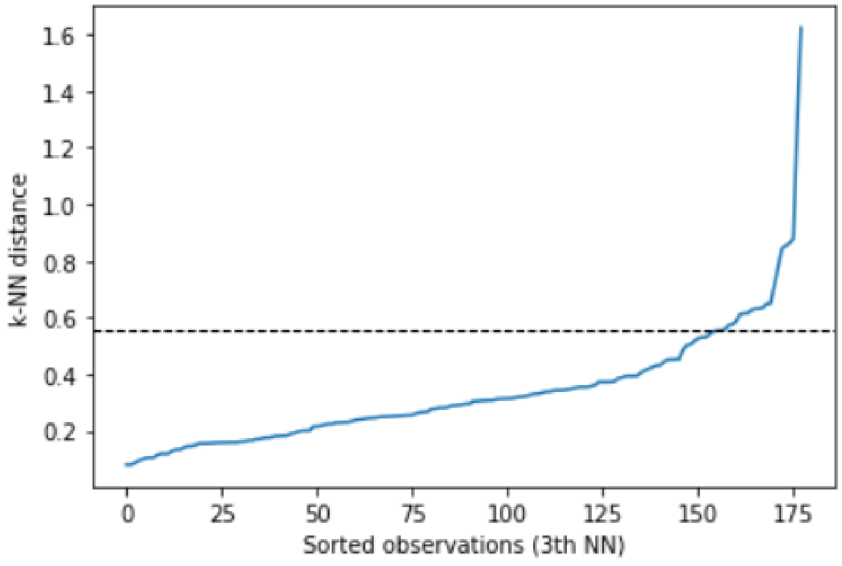


Figure 3

Furthermore, to help choose and further narrow the epsilon value (since we noticed that plotting the dist-graph gives a range), we used an open source python library based on a published paper [6] to implement the kneedle algorithm and find the knee point as shown in Figure 4.

```
Knee Point Value = 0.5539732187671025
```

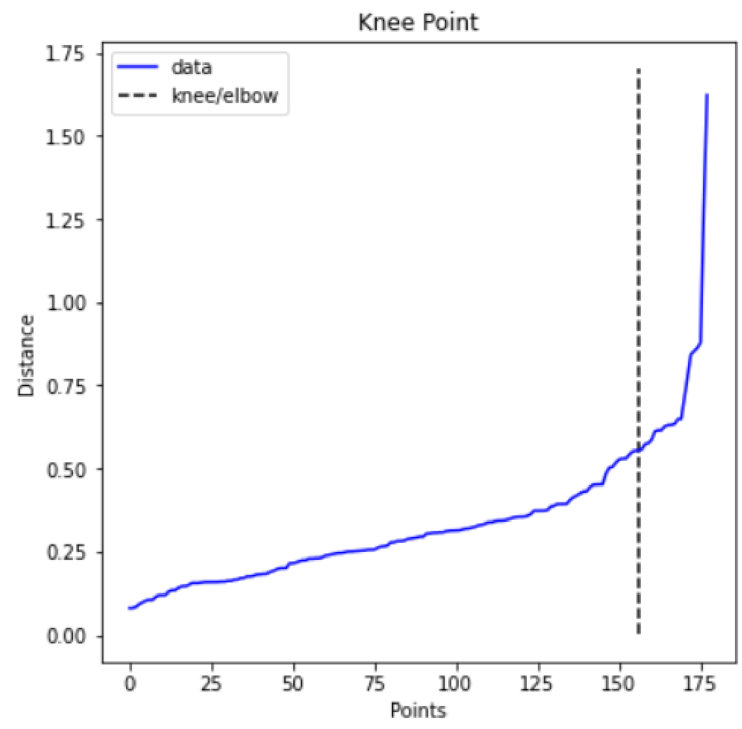


Figure 4

The knee point value is then chosen as the epsilon value for our DBSCAN experiments.

Strength and Weaknesses in DBSCAN

Advantages of DBSCAN :

1. Numbers of clusters do not need to be specified beforehand. (This could be a con too)
2. Performs well with arbitrary shape clusters
3. DBSCAN is robust to outliers and can detect outliers well

Disadvantages of DBSCAN :

1. If there are clusters with varying densities, DBSCAN fails to identify meaningful clusters as the epsilon value is fixed, so it can only identify clusters with a fixed density of points.
2. For DBSCAN, for high dimensional data, it faces the curse of dimensionality as it becomes harder to determine the right epsilon value to use, especially if euclidean distance is used to measure the distance between features of the data.

3. Since the number of clusters does not need to be specified beforehand, for any classification done using clustering, the number of clusters that we get from DBSCAN may not match the number of classes for the dataset.
4. Choosing the right epsilon value is critical in grouping high quality clusters, if not done well, the clusters may be very wrong.

3.2.3 Gaussian Mixture Model Clustering

A Gaussian Mixture Model is a Model based Clustering method that assumes all data points are generated from a mixture of multiple Gaussians with unknown parameters. If the parameters of the Gaussians are known, data points can be easily assigned to clusters and if cluster labels are known, the estimation of the parameters is trivial. The problem lies when both quantities are unknown. To solve this, we make use of the Expectation Maximization Algorithm (EM)

The processes used to generate the data point represents a latent variable, it influences the data but is not observable.

In the EM algorithm, the estimation-step (E-step) would estimate a value for the process latent variable for each data point, and the maximization step (M-step) would optimize the parameters of the probability distributions to best capture the density of the data. The process is repeated until convergence.

If our dataset consists of these observed values X and latent values y and $l(\theta; X, y)$ is the complete data log-likelihood, where θ is the vector of unknown parameters for which we need to find the Maximum Likelihood Estimator (MLE). The EM algorithm is as follows,

E-step: Here, the expected value of $l(\theta; X, y)$ is computed given the observed values X and our current parameter estimate θ^{τ}. We define,

$$Q\left(\theta, \theta^{\tau}\right) = E[\, l(\theta; X, y) \mid X, \theta^{\tau} \,]$$

M-step: Here, the Expectation computed in the E-step is maximized over θ. We set,

$$\theta^{t+1} = \arg arg\, max\, Q\left(\theta, \theta^{\tau}\right)$$

Then we set,

$$\theta^{\tau} = \theta^{t+1}$$

Finally, we increment $t = t + 1$ and repeat the E-step and M-steps until convergence or till the maximum iterations if a value for such has been defined.

Here, we Implement the Gaussian Mixture Model from the sklearn library (https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html).
The model needs an input of how many clusters there are in the data, the max number of iterations to perform for the EM algorithm and Initial parameters. We set the max iterations as 100 and initialize the parameters with random points from the dataset. In datasets where there are true labels associated, we set

the number of clusters to the number of unique labels, otherwise we set the number of clusters based on a performance metric (for example, Silhouette score)

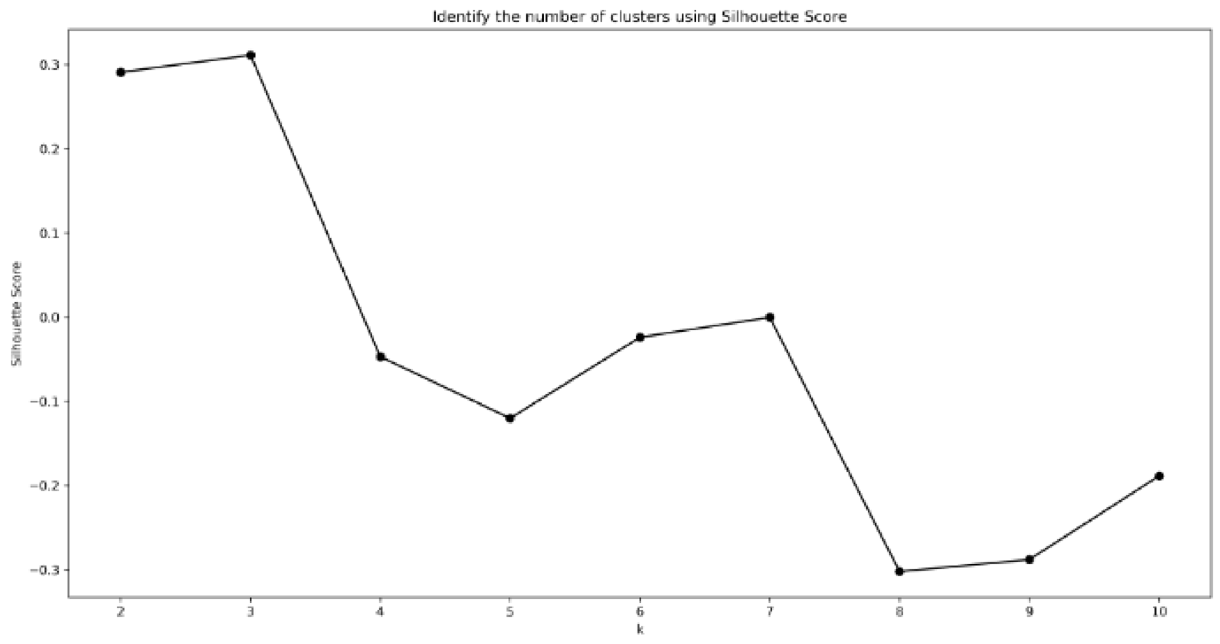


Figure 5

We can then set the clusters according to the best formation i.e the number of clusters corresponding to the highest Silhouette score.

Advantages of using GMM:

GMM models being flexible in terms of cluster co-variance allows them to form non-spherical clusters. GMMs are also a soft clustering method, making them highly useful in cases where each data point need not belong to only one cluster

Disadvantages of using GMM:

GMM is sensitive to the selection of initial parameters. If the parameters selected are random, then the stochastic nature of the algorithm may induce some variation in the results. Moreover there is a possibility of convergence to a local optimum. Also, limiting the EM algorithm to a finite number of steps may reduce its precision.

4. Experiments & Discussion

Metrics	Algorithms	Wine	Customer Spending	Sonar	Yeast
Silhouette score	K-means/++	0.298	0.428	0.126	0.182
	DBSCAN	0.164	0.100	0.277	0.160
	Agglomerative	0.277	0.459	0.248	0.151
	GMM	0.378	0.234	0.154	0.807
Davies-Bouldin Index	K-means/++	1.342	0.825	2.456	1.262
	DBSCAN	3.774	3.633	2.005	3.055
	Agglomerative	1.419	0.836	2.045	1.381
	GMM	0.636	1.296	1.976	0.211
Calinski-harabasz score	K-means/++	77.554	135.102	30.712	316.886
	DBSCAN	5.429	10.402	15.248	15.451
	Agglomerative	67.647	134.347	33.643	273.721
	GMM	317.599	52.349	30.288	39414.908
Classification accuracy	K-means/++	0.978	-	0.457	0.235
	DBSCAN	0.303	-	0.442	0.299
	Agglomerative	0.298	-	0.471	0.409
	GMM	0.702	-	0.514	0.311

Table 1: Metrics for clustering Scaled Dataset **without PCA**

Metrics	Algorithms	Wine	Customer Spending	Sonar	Yeast
Silhouette score	K-means/++	0.587	0.375	0.317	0.309
	DBSCAN	0.167	0.256	0.189	-0.054
	Agglomerative	0.559	0.365	0.448	0.251
	GMM	0.445	0.270	0.300	0.723
Davies-Bouldin Index	K-means/++	0.561	0.858	1.222	0.907
	DBSCAN	2.241	2.683	2.217	2.081
	Agglomerative	0.601	0.845	0.933	0.886
	GMM	0.652	0.984	1.190	0.460
Calinski-harabasz score	K-means/++	401.580	183.407	98.547	785.577
	DBSCAN	83.089	49.576	13.964	17.113
	Agglomerative	341.058	164.807	118.556	700.109
	GMM	124.363	79.290	56.973	3672.241
Classification accuracy	K-means/++	0.978	-	0.442	0.166
	DBSCAN	0.348	-	0.408	0.289
	Agglomerative	0.966	-	0.457	0.134
	GMM	0.174	-	0.543	0.256

Table 2: Metrics for Clustering on Scaled Dataset **with PCA**

4.1 Effects of PCA on Clustering Algorithms

In general, we can see an improvement in all 4 evaluation metric scores after implementing PCA for the wine, sonar and yeast datasets throughout all the clustering algorithms. The reason is that these are datasets with relatively higher dimensions.

Lower dimensions datasets like customer spending on the other hand had a drop in evaluation metric scores across algorithms. This is because of the low dimensionality of the customer spending dataset, resulting in a significant amount of information loss after PCA is implemented, compared to the other datasets with high dimensionality.

Another observation that can be made is that for distance based clustering algorithms like K-means, DBSCAN and Agglomerative Clustering, the effects of PCA are more significant as compared to distribution based algorithms like GMM.

PCA helps distance based algorithms, especially euclidean distance because in higher dimensions, the distance becomes distorted as the ratio of the nearest and furthest point approaches 1, and the point becomes universally distant to each other [19], making clustering very hard as we are unable to identify similarity and differences between points.

Lastly, for GMM, we observed that PCA can still improve performance albeit lesser than distance based and the reason is that the EM algorithm in GMM may fail to converge in higher dimensionality datasets such as sonar dataset

In the case of the Yeast dataset,The high values for silhouette score (0.807) and low values for Davies-Bouldin Index (0.211) indicate that there are dense clusters formed which are well separated with lesser overlap between the clusters. This is observed most likely as the structure of the data could be hard to interpret for partition and density based clustering methods, but the EM still manages to converge and in this case, It was likely an optimal convergence.

We can conclude from this that distance based algorithms using euclidean distance benefits more from PCA.

4.2 Discussion of clustering algorithms on datasets with PCA

4.2.1 Wine Dataset

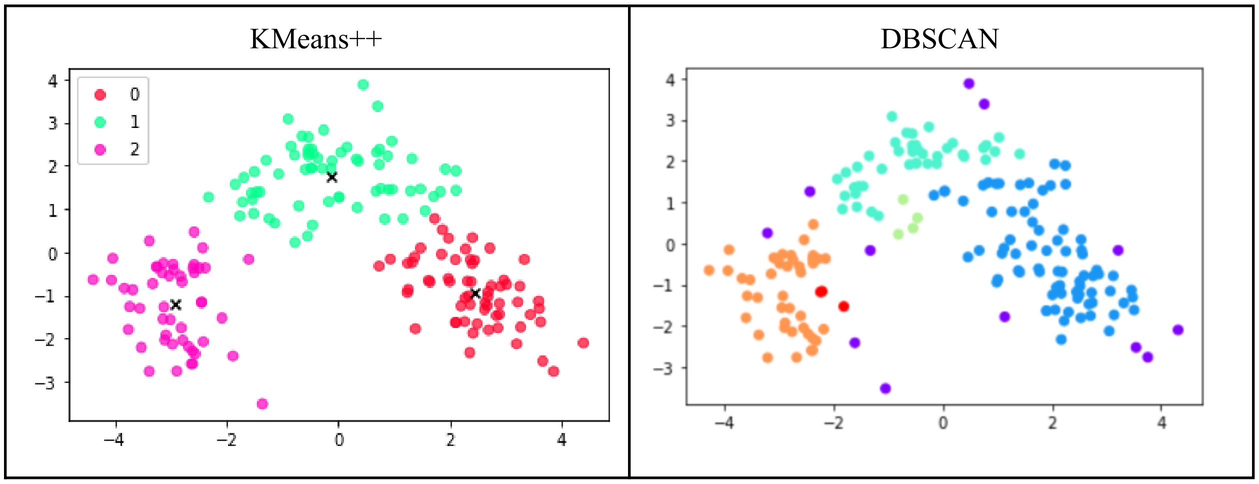
From all four metrics, K-means++ performed the best, with agglomerative clustering close behind for all metrics. For the worst algorithm, DBSCAN performed the worst.

As shown below, both KMeans++ and Agglomerative Clustering have the best formed clusters. However, even though DBSCAN has the worst score for all four metrics, it still clustered better visually.

From this dataset, we can see that clusters are in spherical shape, which k-means is good for clustering.

However, although DBSCAN scores are the worst, it is useful for helping identify outliers (as denoted by the purple points) that other algorithms are unable to detect. We can also see that even though not specifying the number of clusters should be an advantage of DBSCAN, but for datasets with labels which has predefined clusters/classification (e.g. 3 in wine dataset), not being able to specify the number of clusters becomes a disadvantage as it will be wrongly classified.

Because of this, we can also draw conclusions that the reason why k-means++ and agglomerative, which are sensitive to outliers, had a good score was because there were not many outliers.



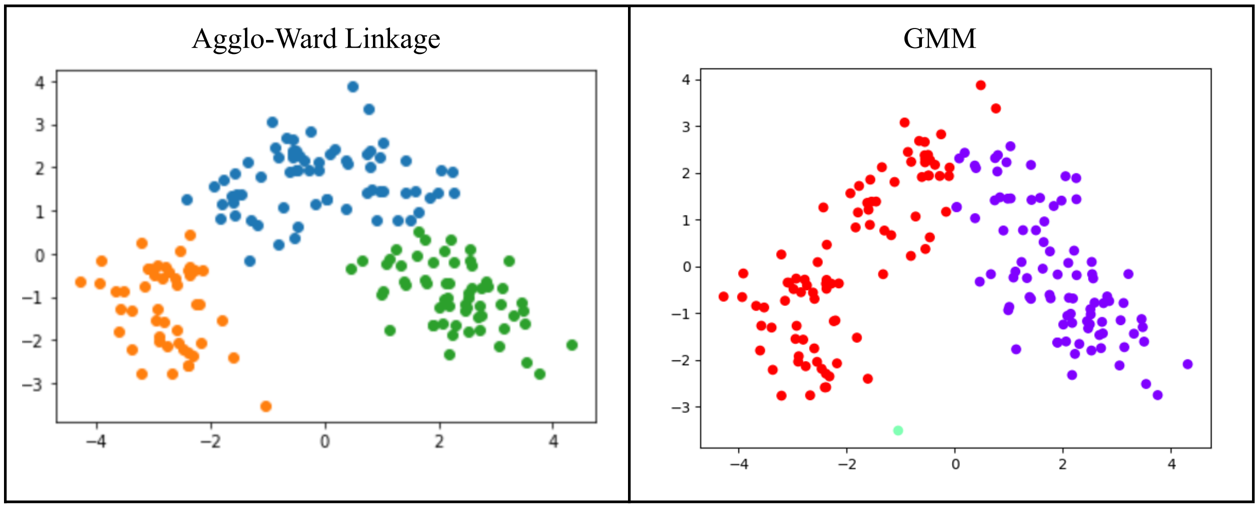
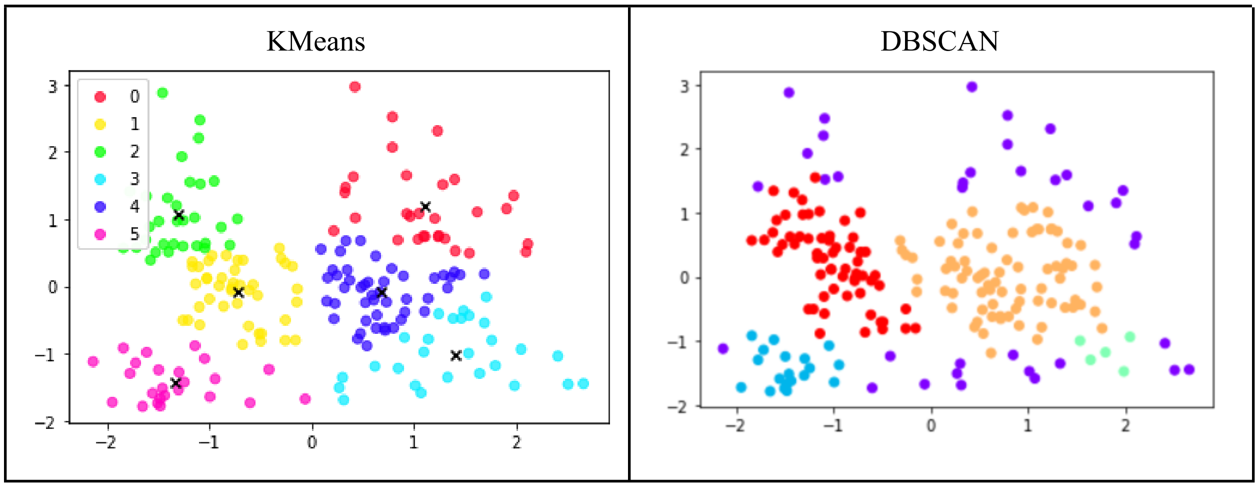


Figure 6

4.2.2 Customer Spending Dataset

K-means and agglomerative did the best and had roughly the same scores for the 3 different metrics (as there are no labels, so no classification accuracy).
DBSCAN had the worst scores for the three metrics. However, it is useful for helping identify outliers (as denoted by the purple points) that other algorithms are unable to detect. However, visually, we can see it forms the worst clusters.



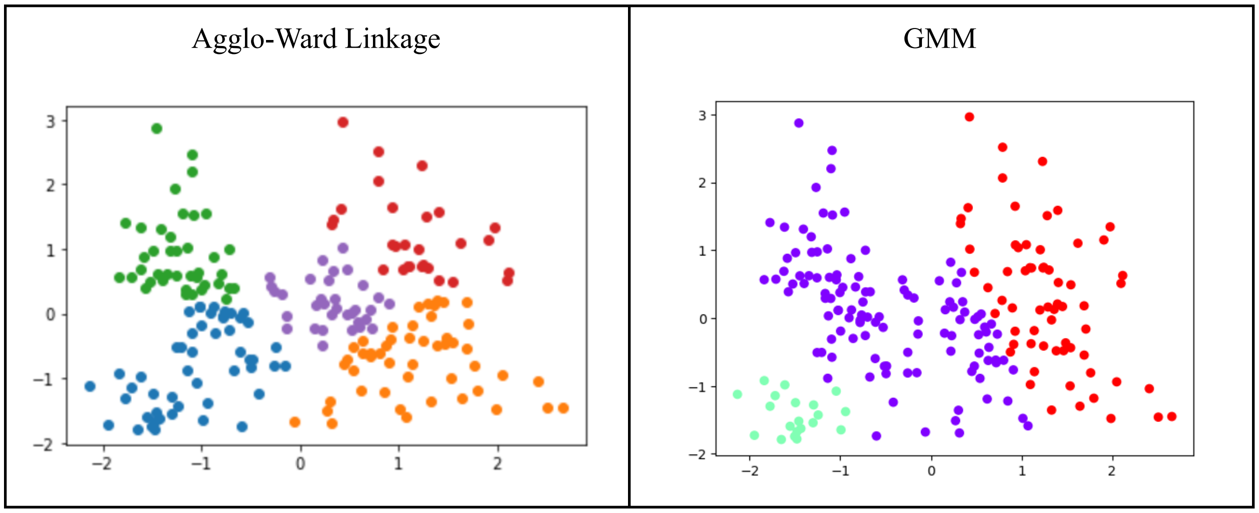


Figure 7

4.2.3 Sonar Dataset

Agglomerative Clustering did the best for this dataset, with GMM and K-means++ performing slightly worse for the score metric (and having similar metric scores) compared to agglomerative clustering.

DBSCAN performed the worst overall out of all the algorithms for the score metric. However, as mentioned, it is useful for helping identify outliers (as denoted by the purple points) that other algorithms are unable to detect.

However, visually, we can see that GMM and agglomerative looked the same visually and performed a better job in clustering compared to KMeans++. The reason that attributed to k-means poor visualization is that k-means only performs well with spherical clusters

Moreover, DBSCAN is also unable to form correct clusters because of the varying density of the data points. As seen from the figure below, the right half of the data points are more sparse as in comparison to the denser left half, and since the epsilon value of DBSCAN is fixed prior to the clustering, it is unable to identity the cluster on the right with a different density.

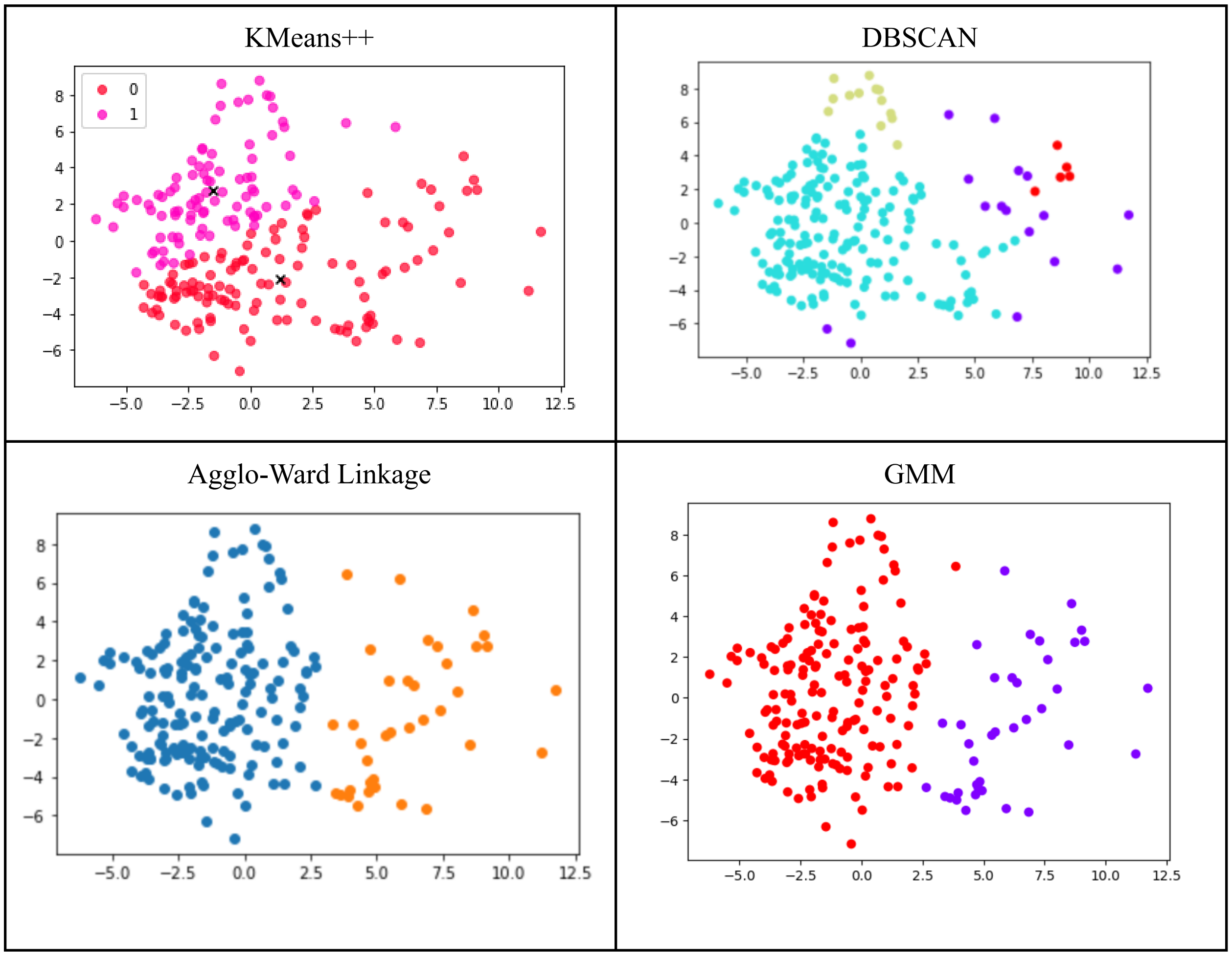


Figure 8

4.2.4 Yeast Dataset

GMM did the best for the Yeast dataset, followed by Kmeans++ and Agglomerative Clustering close behind according to the evaluation metric scores.

DBSCAN did the worst overall out of all the algorithms for the score metric. However, as mentioned, it is useful for helping identify outliers (as denoted by the purple points) that other algorithms are unable to detect.

Visually, DBSCAN did the worst as we can see from the figure below, this is largely because all the points are dense and close together. DBSCAN clusters based on density thus, it would treat all the dense points as one big cluster.

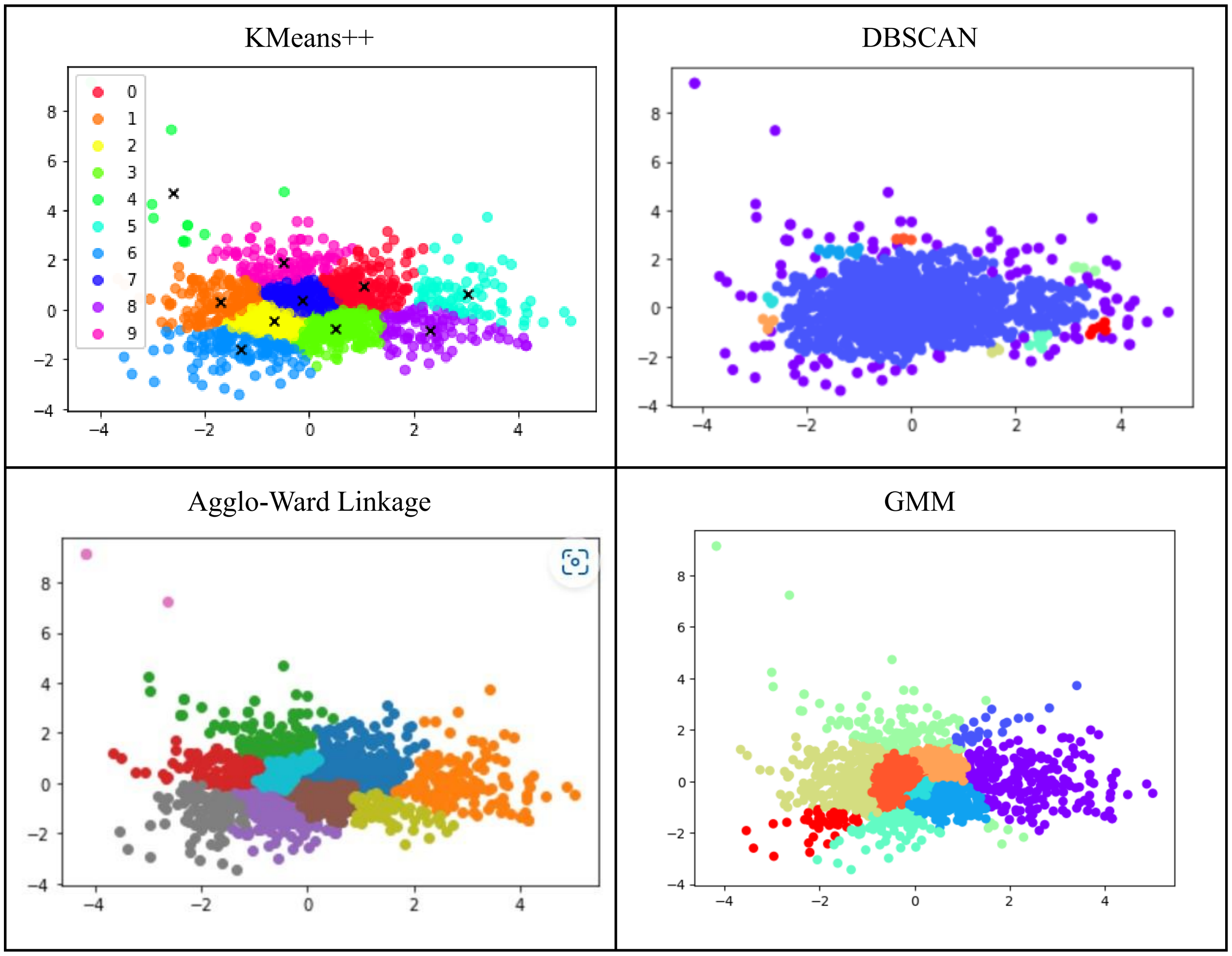


Figure 9

5. Conclusion

This report provided a comprehensive study of clustering algorithms belonging to different categories as defined in previous literature.

From our obtained results, we can see that no algorithm provides a “one size fits all” solution. Different algorithms outperform each other when varying the dataset. We provided empirical results to justify competitive performance between various algorithms, and future work should be dedicated to better interpretation of the performance metrics for clustering algorithms.

Circling back to the datasets that we chose, we have seen different results:

- Data set that has was not a classification problem (customer): KMeans and Agglomerative performed the best
- Data set that has a high dimensionality (sonar): Agglomerative outperformed the rest
- Large data set (yeast): GMM outperformed the rest
- Simple data set (wine): KMeans++ and Agglomerative performed the best

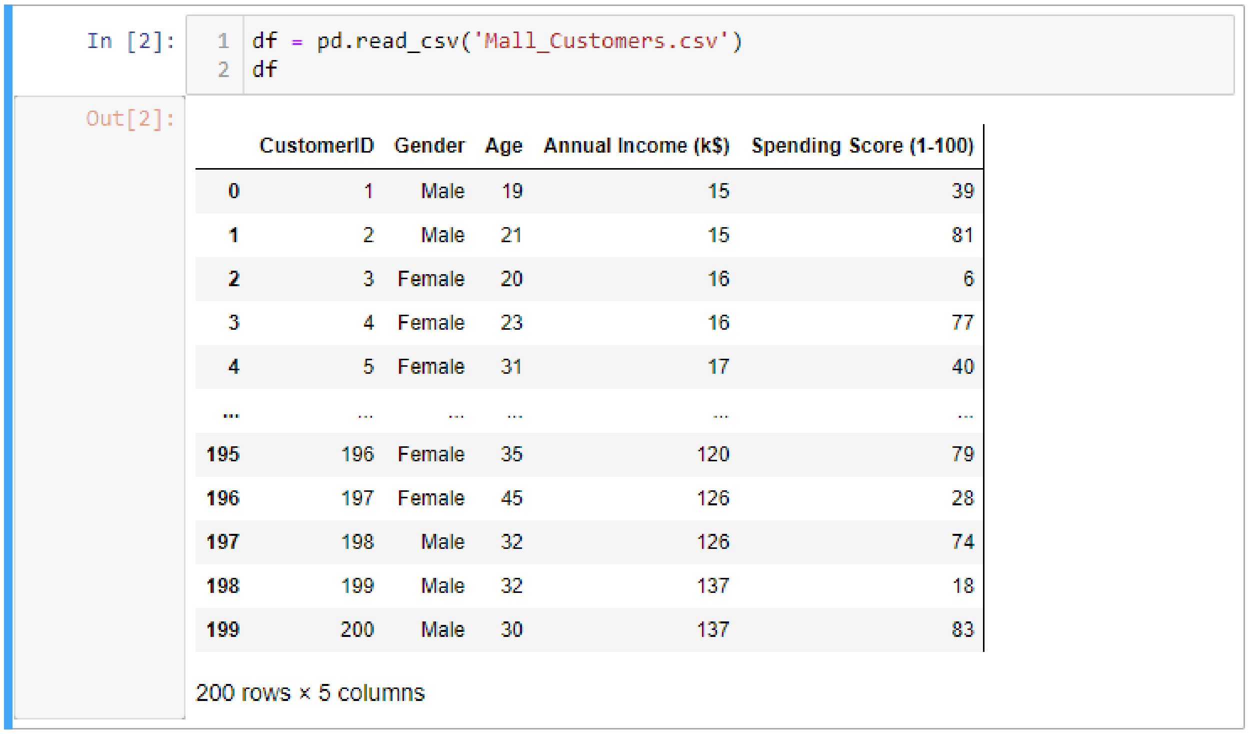
In particular, we studied how higher dimensions datasets negatively impact distance based clustering algorithms, and studied the effects of PCA to reduce dimensionality of such datasets and our findings revealed that there was indeed gain in metric scores for the different algorithms.

Moreover, more studies are needed on the impact of initial parameters on cluster formation. The selection of type of linkage or distance metrics in the case of Agglomerative clustering, and the selections of initial parameters or type of covariance parameters with the shape of the data distribution unknown.

Appendix A

Example of data pre-processing:

Initial dataset -



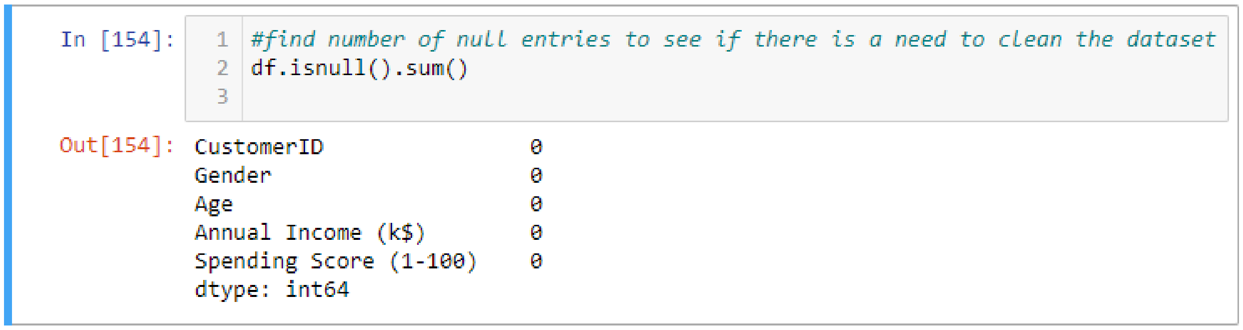
```
In [2]:  df = pd.read_csv('Mall_Customers.csv')
          df
```

Out[2]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

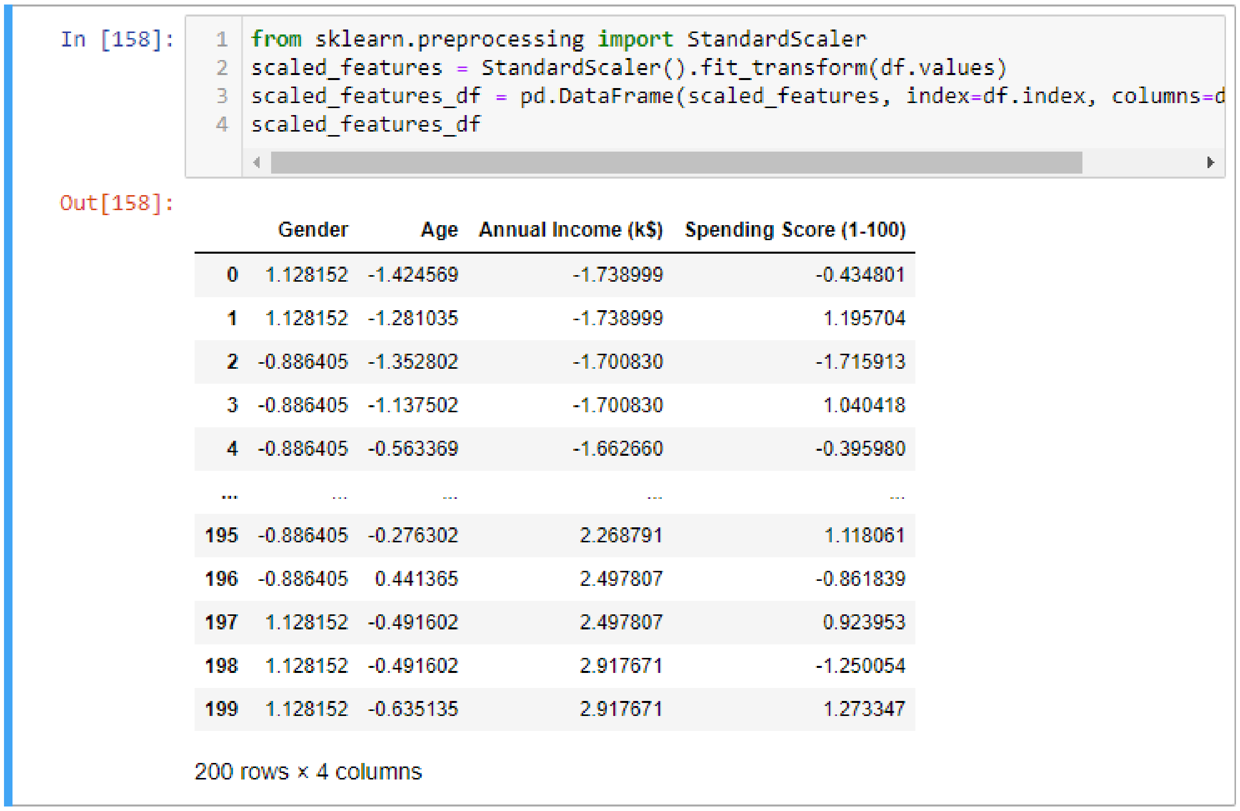
Check for null values to determine if there is a need to remove rows



```
In [154]:  #find number of null entries to see if there is a need to clean the dataset
           df.isnull().sum()

Out[154]:  CustomerID                0
           Gender                    0
           Age                       0
           Annual Income (k$)        0
           Spending Score (1-100)    0
           dtype: int64
```

Perform standardization to reduce impacts of anomalies on the model



```
In [158]: from sklearn.preprocessing import StandardScaler
          scaled_features = StandardScaler().fit_transform(df.values)
          scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=d
          scaled_features_df
```

Out[158]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.738999	-0.434801
1	1.128152	-1.281035	-1.738999	1.195704
2	-0.886405	-1.352802	-1.700830	-1.715913
3	-0.886405	-1.137502	-1.700830	1.040418
4	-0.886405	-0.563369	-1.662660	-0.395980
...	...	...	...	...
195	-0.886405	-0.276302	2.268791	1.118061
196	-0.886405	0.441365	2.497807	-0.861839
197	1.128152	-0.491602	2.497807	0.923953
198	1.128152	-0.491602	2.917671	-1.250054
199	1.128152	-0.635135	2.917671	1.273347

200 rows × 4 columns

Appendix B

Comparison of evaluation metric score on different linkage methods on Pre-PCA Dataset

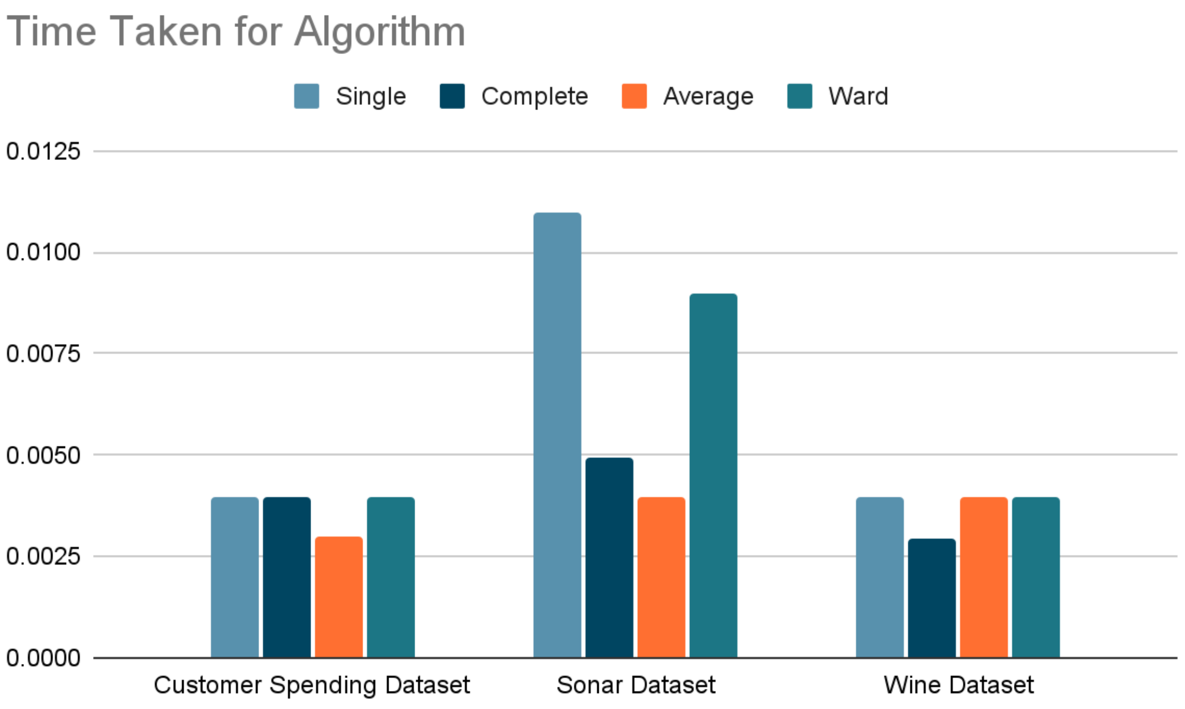
		Single	Complete	Average	Ward
Customer Spending Dataset	Time-taken	0.00399	0.00399	0.00299	0.00399
	Silhouette score	0.22254	0.42315	0.38017	0.45904
	Davies Bouldin score	0.51965	0.89742	0.89558	0.83633
	Calinski Harabasz score	11.52832	165.61825	132.48512	134.34740
	Classification Accuracy	-	-	-	-
Sonar Dataset	Time-taken	0.01097	0.00496	0.00399	0.00898
	Silhouette score	0.37431	0.23978	0.37433	0.24824
	Davies Bouldin score	0.48399	2.10712	0.48391	2.04545
	Calinski Harabasz score	3.91306	33.41459	3.9133	33.64336
	Classification Accuracy	0.52885	0.47596	0.52885	0.47115
Wine Dataset	Time-taken	0.00399	0.00296	0.00399	0.00399

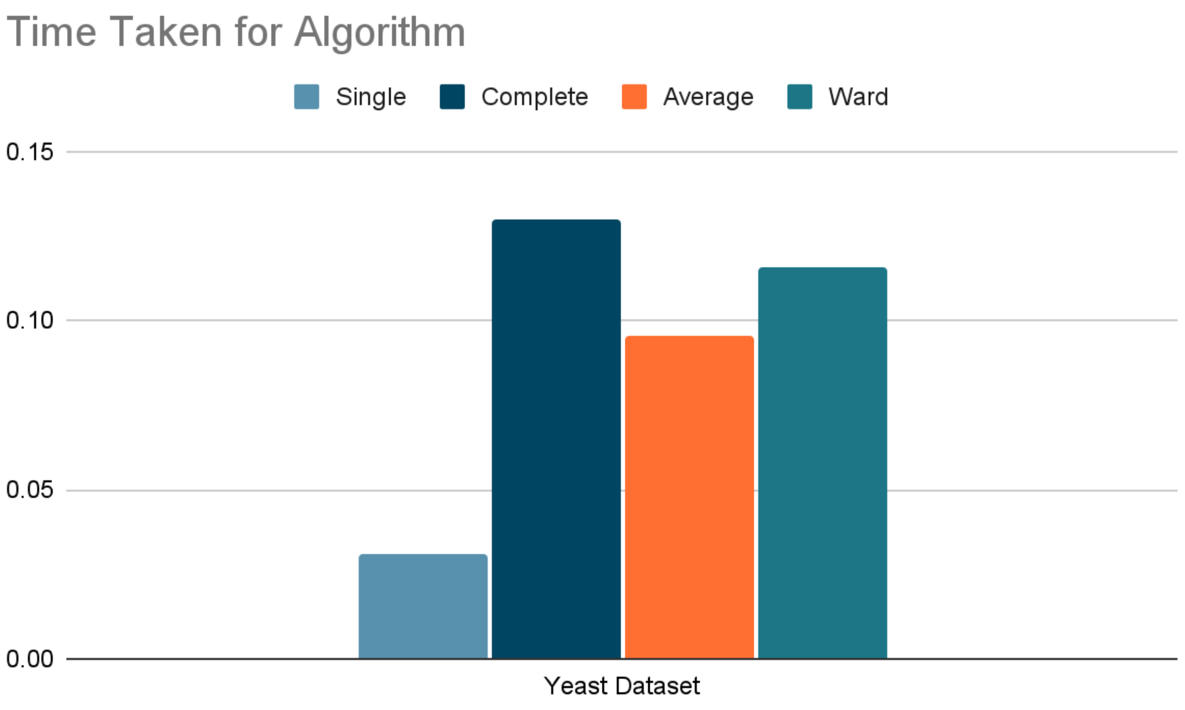
	Silhouette score	0.18274	0.20379	0.15753	0.27744
	Davies Bouldin score	0.91052	1.89610	1.02989	1.41859
	Calinski Harabasz score	4.06166	48.98983	4.03144	67.64747
	Classification Accuracy	0.37640	0.41573	0.38764	0.29775
Yeast Dataset	Time-taken	0.03092	0.12969	0.09574	0.11569
	Silhouette score	0.38113	0.11051	0.20716	0.12830
	Davies Bouldin score	0.36874	1.35933	0.80695	1.62001
	Calinski Harabasz score	11.75358	148.38754	39.82570	199.83605
	Classification Accuracy	0.30997	0.26078	0.34097	0.16712

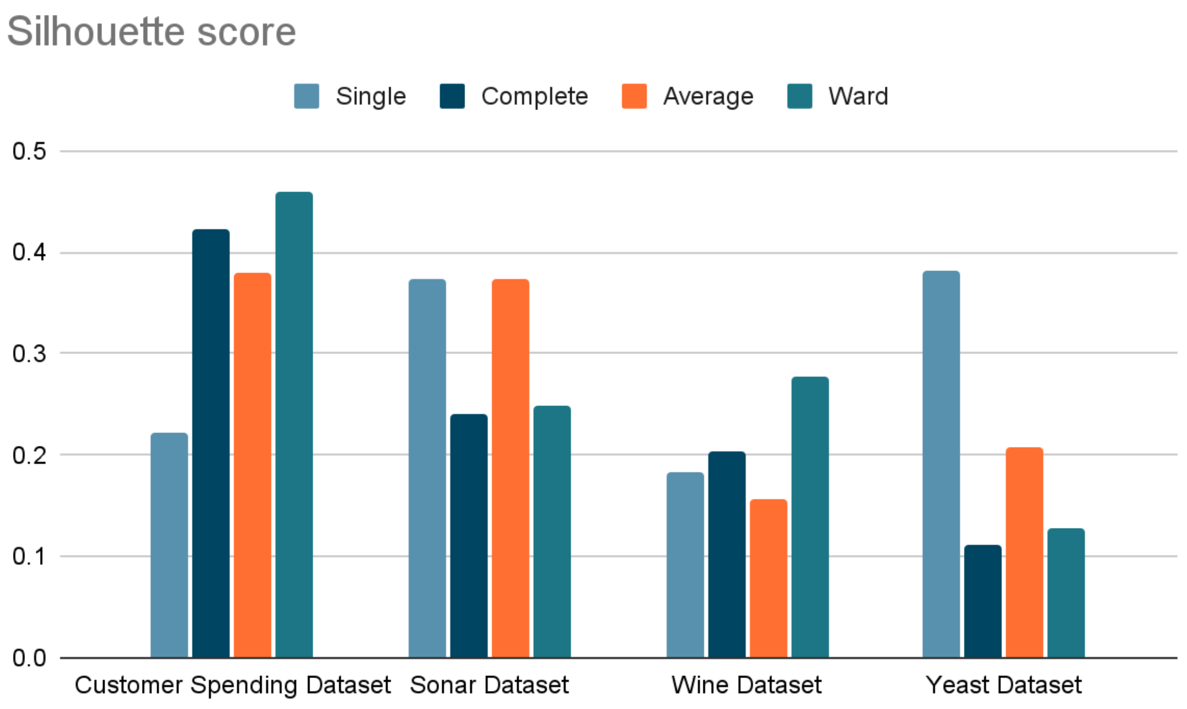
Comparison of evaluation metric score on different linkage methods on Post-PCA Dataset

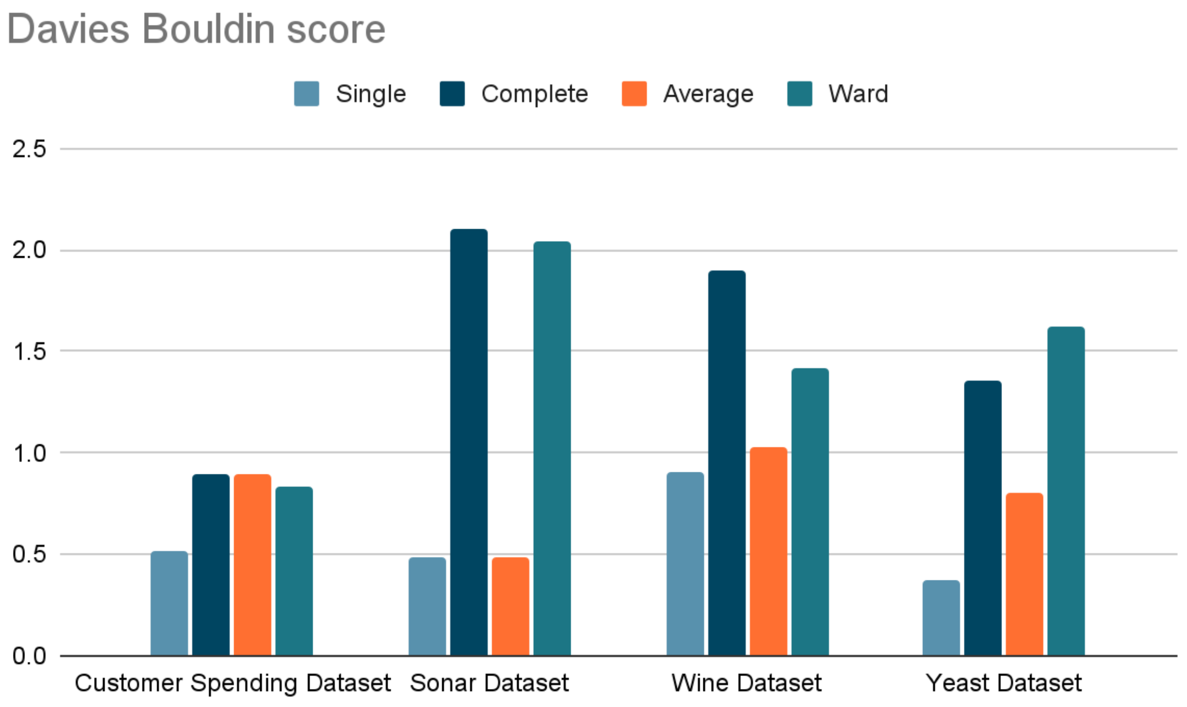
		Single	Complete	Average	Ward
Customer Spending Dataset	Time-taken (s)	0.00199	0.00399	0.00299	0.00299
	Silhouette score	0.06108	0.37390	0.34589	0.36494
	Davies Bouldin score	0.52834	0.92477	0.69702	0.84466
	Calinski Harabasz score	4.1151	158.91058	116.03522	164.80663
	Classification Accuracy	-	-	-	-
Sonar Dataset	Time-taken	0.00299	0.00298	0.00199	0.00299
	Silhouette score	0.45503	0.44830	0.44404	0.44830
	Davies Bouldin score	0.36452	0.93313	0.89852	0.93313
	Calinski Harabasz score	5.9912	118.55585	115.04512	118.55585
	Classification Accuracy	0.52885	0.45673	0.45673	0.45673
Wine Dataset	Time-taken	0.00200	0.00199	0.00299	0.00299

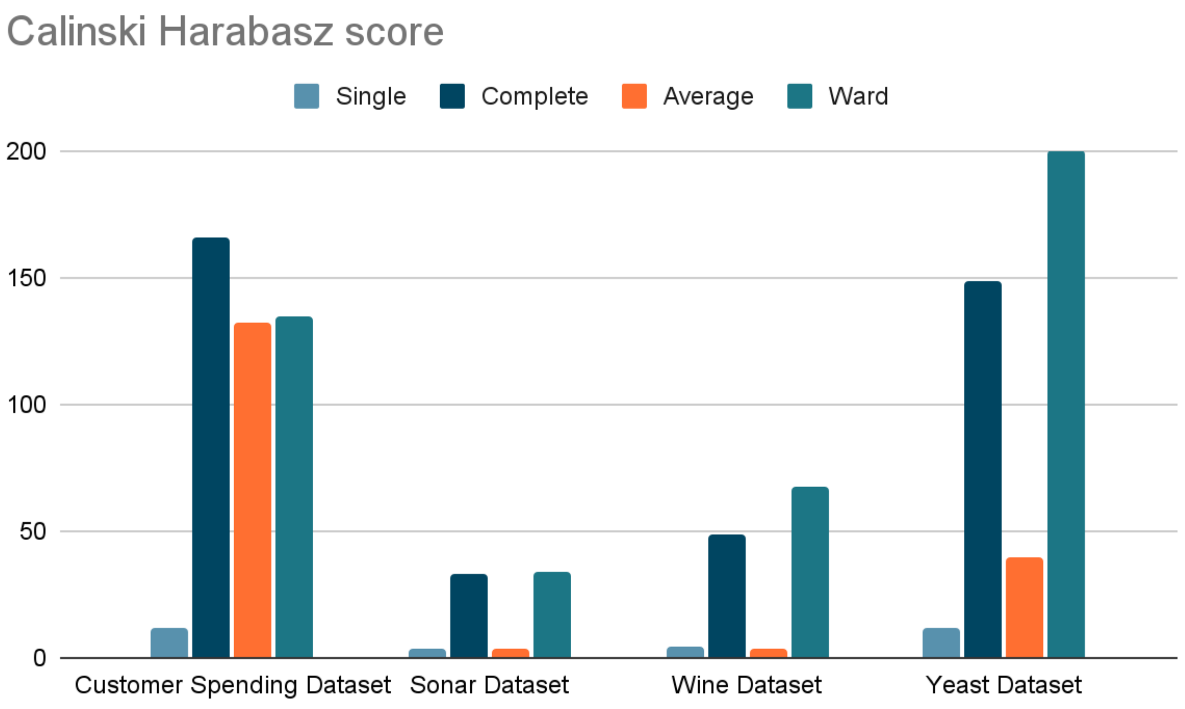
	Silhouette score	0.01915	0.51355	0.55421	0.55909
	Davies Bouldin score	0.63204	0.62684	0.60265	0.60134
	Calinski Harabasz score	2.57756	277.35405	333.53167	341.05824
	Classification Accuracy	0.41011	0.43820	0.33146	0.96629
Yeast Dataset	Time-taken	0.03391	0.11270	0.09475	0.12666
	Silhouette score	0.28647	0.22672	0.22267	0.25141
	Davies Bouldin score	0.32794	0.92899	0.64373	0.93721
	Calinski Harabasz score	18.73777	486.57366	251.30367	686.75408
	Classification Accuracy	0.30997	0.21968	0.26482	0.17049

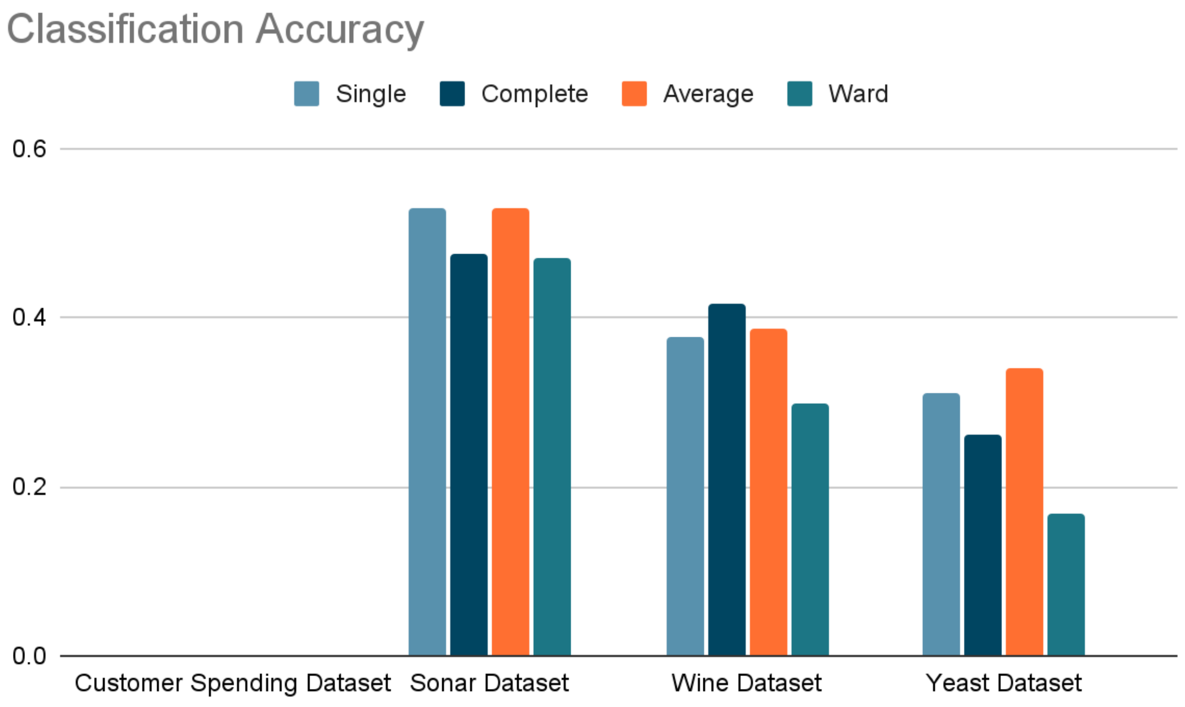

Time Taken for Algorithm
Single
Complete
Average
Ward
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Customer Spending Dataset
Sonar Dataset
Wine Dataset


Time Taken for Algorithm
Single
Complete
Average
Ward
0.15
0.10
0.05
0.00
Yeast Dataset


Silhouette score
Single
Complete
Average
Ward
0.5
0.4
0.3
0.2
0.1
0.0
Customer Spending Dataset
Sonar Dataset
Wine Dataset
Yeast Dataset


Davies Bouldin score
Single
Complete
Average
Ward
2.5
2.0
1.5
1.0
0.5
0.0
Customer Spending Dataset
Sonar Dataset
Wine Dataset
Yeast Dataset


Calinski Harabasz score
Single
Complete
Average
Ward
200
150
100
50
0
Customer Spending Dataset
Sonar Dataset
Wine Dataset
Yeast Dataset


Classification Accuracy
Single
Complete
Average
Ward
0.6
0.4
0.2
0.0
Customer Spending Dataset
Sonar Dataset
Wine Dataset
Yeast Dataset

Citation

1. D. Xu and Y. Tian, “A comprehensive survey of clustering algorithms - annals of data science,” *SpringerLink*, 12-Aug-2015. [Online]. Available: https://link.springer.com/article/10.1007/s40745-015-0040-1. [Accessed: 15-Oct-2022].

2. A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: A Review,” *IEEE Xplore*, Jan-2000. [Online]. Available: https://ieeexplore.ieee.org/document/824819. [Accessed: 15-Oct-2022].

3. A. K. Mann and N. Kaur, “Review Paper on Clustering Techniques,” *Global Journal of Computer Science and Technology Software & Data Engineering*, vol. 13, no. 5. 2013.

4. M. G. H. Omran, A. P. Engelbrecht, and A. Salman, “An overview of clustering methods,” *Intelligent Data Analysis*, vol. 11, no. 6, pp. 583–605, 2007.

5. Mahalanobis, Prasanta Chandra (1936). "On the generalised distance in statistics" (PDF). Proceedings of the National Institute of Sciences of India. 2 (1): 49–55.

6. Stephanie, “Mahalanobis distance: Simple definition, examples,” *Statistics How To*, 22-Sep-2020. [Online]. Available: https://www.statisticshowto.com/mahalanobis-distance/. [Accessed: 15-Oct-2022].

7. A. Bhardwaj, “Silhouette coefficient : Validating clustering techniques,” *Medium*, 27-May-2020. [Online]. Available: https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c. [Accessed: 15-Oct-2022].

8. PyShark, “Davies-Bouldin index for k-means clustering evaluation in python: Python-bloggers,” *Python-bloggers*, 02-Jun-2021. [Online]. Available: https://python-bloggers.com/2021/06/davies-bouldin-index-for-k-means-clustering-evaluation-in-python/. [Accessed: 15-Oct-2022].

9. H. Wei, “How to measure clustering performances when there are no ground truth?,” *Medium*, 02-Jan-2020. [Online]. Available: https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c#:~:text=The%20Calinski%2DHarabasz%20index%20also,score%20%2C%20the%20better%20the%20performances. [Accessed: 15-Oct-2022].

10. A. Parker and T. Jones, “What is curse of dimensionality in machine learning? Explained - Machine learning pro,” *Machine Learning Pro - Addressing world problems through Artificial Intelligence.*, 24-Aug-2022. [Online]. Available: https://www.machinelearningpro.org/curse-of-dimensionality-in-machine-learning/. [Accessed: 15-Oct-2022].

11. N. Raj, “The curse of dimensionality and its cure,” *Medium*, 25-Apr-2021. [Online]. Available: https://medium.com/analytics-vidhya/the-curse-of-dimensionality-and-its-cure-f9891ab72e5c. [Accessed: 15-Oct-2022].

12. I. T. Jolliffe and J. Cadima, “Principal component analysis: A review and recent developments,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 13-Apr-2016. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202. [Accessed: 15-Oct-2022].

13. I. Dabbura, “K-means clustering: Algorithm, applications, evaluation methods, and drawbacks,” *Medium*, 27-Sep-2022. [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a#:~:text=Kmeans%20Algorithm,belongs%20to%20only%20one%20group. [Accessed: 16-Oct-2022].

14. “Partitional(K-means), hierarchical, density-based (DBSCAN).” [Online]. Available: https://cs.wmich.edu/alfuqaha/summer14/cs6530/lectures/ClusteringAnalysis.pdf. [Accessed: 16-Oct-2022].

15. P. Sharma, “K means clustering: K means clustering algorithm in Python,” *Analytics Vidhya*, 15-Jun-2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/. [Accessed: 16-Oct-2022].

16. “K-means advantages and disadvantages | machine learning | google developers,” *Google*, 18-Jul-22AD. [Online]. Available: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages. [Accessed: 16-Oct-2022].

17. S. Yıldırım, “DBSCAN clustering‐explained,” *Medium*, 22-Apr-2020. [Online]. Available: https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556. [Accessed: 16-Oct-2022].

18. “Sklearn.cluster.DBSCAN,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html. [Accessed: 16-Oct-2022].

19. Aggarwal, Charu & Hinneburg, Alexander & Keim, Daniel. (2002). On the Surprising Behavior of Distance Metric in High-Dimensional Space. First publ. in: Database theory, ICDT 200, 8th International Conference, London, UK, January 4 - 6, 2001 / Jan Van den Bussche ... (eds.). Berlin: Springer, 2001, pp. 420-434 (=Lecture notes in computer science ; 1973).

20. Banerjee, A., Shan, H. (2011). Model-Based Clustering. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_554